Летняя школа по компьютерным наукам – 2016

НИУ ВШЭ, Центр «Стратегия»

Параллель B, занятие 1: Сортировки и их применение

Автор: Михаил Густокашин, Павел Шевчук

Сортировка

Под упорядоченным массивом будем понимать массив, упорядоченный по неубыванию, т.е. a[1] ≤ a[2] ≤ … ≤ a[N].

У нас имеется заданная своими границами область поиска. Мы выбираем ее середину. Если искомый элемент меньше чем средний, то поиск осуществляется в левой части, иначе – в правой. Действительно, если искомый элемент меньше среднего, то и меньше всех элементов, которые находятся правее среднего, а значит, их сразу можно исключить из рассмотрения. Аналогично для случая, когда искомый элемент больше среднего.

Код, осуществляющий бинарный поиск в упорядоченном массиве выглядит так:

```
long long left = 0, right = n;
while (left < right) {
        long long m = (left + right) / 2;
        if (x >= a[m])
                right = m;
        else
                left = m + 1;
}
```

std::lower_bound(begin_iterator, end_iterator, value) - левый бинарный поиск
Эта функция находит между begin и end (begin включая, end не включая) самый левый элемент x, такой что x не меньше value. Если же этого элемента нет, то он возвращает end_iterator (как будто есть ещё элементы).
std::upper_bound(begin_iterator, end_iterator, value) - правый бинарный поиск
Эта функция находит между begin и end (begin включая, end не включая) самый левый элемент x, такой что x строго больше value. Если же этого элемента нет, то он возвращает end_iterator (как будто есть ещё элементы).

Обе функции возвращают итератор.

В сумме эти две функции используются так:

std::vector<int> v;
//...
lower_bound(v.begin(), v.end(), x), upper_bound(v.begin(), v.end(), x) - полуинтервал элементов, равных x

Например:
Позиция элемента, равного x в векторе v
std::lower_bound(v.begin(), v.end(), x) - v.begin()
Количество элементов, равных x
cout << std::upper_bound(v.begin(), v.end(), x) - std::lower_bound(v.begin(), v.end(), x);
Чтобы вывести наименьший элемент больше либо равный заданному, необходимо написать:
cout << *std::lower_bound(v.begin(), v.end(), x);

Бинарный поиск по ответу

Во многих задачах в качестве ответа необходимо вывести какое-либо число. При этом достаточно легко сказать, больше ли это число, чем нужно, или меньше, несмотря на то, что вычисление самого ответа может быть довольно трудоемкой операцией. В таком случае мы можем выбрать число заведомо меньшее ответа и число заведомо большее ответа, а правильное решение искать бинарным поиском.
При реализации бинарного поиска по ответу удобно использовать функцию good, которая возвращает истину, если ответ подходит и ложь в противном случае. Здесь приведен пример, когда до какого-то момента функция плохая, а начиная с него – функция хорошая. Код бинпоиска не отличается от задачи к задаче.

```
#include <iostream>

using namespace std;

bool good(long long days, long long n, long long k) {
	return days * (2 * k + days - 1) / 2 >= n;
}

int main() {
	int n, k;
	cin >> n >> k;
```

```
	long long left = 0, right = n;
	while (left < right) {
		long long m = (left + right) / 2;
		if (good(m, n, k))
			right = m;
		else
			left = m + 1;
	}
	cout << left;
	return 0;
}
```